Partial LLL Reduction

Xiaohu Xie
School of Computer Science
McGill University
Montreal, Quebec, Canada H3A 2A7
Email: xiaohu.xie@mail.mcgill.ca

Xiao-Wen Chang
School of Computer Science
McGill University
Montreal, Quebec, Canada H3A 2A7
Email: chang@cs.mcgill.ca

Mazen Al Borno
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 2E4
Email: mazen@dgp.toronto.edu

Abstract—The Lenstra-Lenstra-Lovasz (LLL) reduction has wide applications in digital communications. It can greatly improve the speed of the sphere decoding (SD) algorithms for solving an integer least squares (ILS) problem and the performance of the Babai integer point, a suboptimal solution to the ILS problem. Recently Ling and Howgrave-Graham proposed the so-called effective LLL (ELLL) reduction. It has less computational complexity than LLL, while it has the same effect on the performance of the Babai integer point as LLL. In this paper we propose a partial LLL (PLLL) reduction. PLLL avoids the numerical stability problem with ELLL, which may result in very poor performance of the Babai integer point. Furthermore, numerical simulations indicated that it is faster than ELLL. We also show that in theory PLLL and ELLL have the same effect on the search speed of a typical SD algorithm as LLL.

I. Introduction

In a multiple-input and multiple-output (MIMO) system, often we have the following linear model:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v}, \tag{1}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is the channel output vector, $\boldsymbol{v} \in \mathbb{R}^n$ is the noise vector following a normal distribution $\mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$, $\boldsymbol{H} \in \mathbb{R}^{n \times m}$ is the channel matrix, and $\boldsymbol{x} \in \mathbb{Z}^m$ is the unknown integer data vector. In some applications where complex ILS problems may need to be solved instead, we can first transform the complex ILS problems to equivalent real ILS problems. For simplicity, like [1], in this paper we assume $m = n$ and $\boldsymbol{H}$ is nonsingular.

To estimate $\boldsymbol{x}$, one solves an integer least squares problem

$$\min_{\boldsymbol{x} \in \mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|_2^2, \tag{2}$$

which gives the maximum-likelihood estimate of $\boldsymbol{x}$. It has been proved that the ILS problem is NP-hard [2]. For applications which have high real-time requirement, an approximate solution of (2) is usually computed instead. A often used approximation method is the nearest plane algorithm proposed by Babai [3] and the produced approximate integer solution is referred to as the Babai integer point. In communications, a method for finding this approximate solution is referred to as a successive interference cancellation decoder.

A typical method to solve (2) is a sphere decoding (SD) algorithm, such as the Schnorr-Euchner algorithm (see [4] and [5]) or its variants (see, e.g., [6] and [7]). A SD algorithm has two phases. First the reduction phase transforms (2) to an equivalent problem. Then the search phase enumerates

integer points in a hyper-ellipsoid to find the optimal solution. The reduction phase makes the search phase easier and more efficient. The Lenstra-Lenstra-Lovasz (LLL) reduction [8] is the mostly used reduction in practice. An LLL reduced basis matrix has to satisfy two conditions. One is the size-reduction condition and the other is the Lovasz condition (see Section II for more details). Recently Ling and Howgrave-Graham [1] argued geometrically that the size-reduction condition does not change the performance of the Babai integer point. Then they proposed the so-called effective LLL reduction (to be referred to as ELLL) which mostly avoids size reduction. They proved that their ELLL algorithm has less time complexity than the original LLL algorithm given in [8]. However, as implicitly pointed out in [1], the ELLL algorithm has a numerical stability problem. Our simulations, presented in Section V, will indicate that ELLL may give a very bad estimate of $\boldsymbol{x}$ than the LLL reduction due to its numerical stability problem.

In this paper, we first show algebraically that the size-reduction condition of the LLL reduction has no effect on a typical SD search process. Thus it has no effect on the performance of the Babai integer point, the first integer point found in the search process. Then we propose a partial LLL reduction algorithm, to be referred to as PLLL, which avoids the numerical stability problem with ELLL. Numerical simulations indicate that it is faster than ELLL and is as numerically stable as LLL.

II. LLL Reduction

In matrix language, the LLL reduction can be described as a QRZ factorization [9]:

$$\boldsymbol{Q}^T \boldsymbol{H} \boldsymbol{Z} = \boldsymbol{R}, \tag{3}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ is orthogonal, $\boldsymbol{Z} \in \mathbb{Z}^{n \times n}$ is a unimodular matrix (i.e., $\det(\boldsymbol{Z}) = \pm 1$), and $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ is upper triangular and satisfies the following two conditions:

$$|r_{i,j}| \leq |r_{i,i}|/2, \quad 1 \leq i < j \leq n$$
$$\delta \, r_{i-1,i-1}^2 \leq r_{i-1,i}^2 + r_{i,i}^2, \quad 1 < i \leq n, \tag{4}$$

where the parameter $\delta \in (1/4, 1]$. The first condition in (4) is the size-reduction condition and the second condition in (4) is the Lovasz condition.

Define $\bar{\boldsymbol{y}} = \boldsymbol{Q}^T \boldsymbol{y}$ and $\boldsymbol{z} = \boldsymbol{Z}^{-1} \boldsymbol{x}$. Then it is easy to see that the ILS problem (2) is reduced to

$$\min_{\boldsymbol{z} \in \mathbb{Z}^n} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2. \tag{5}$$

If $\hat{z}$ is the solution of the reduced ILS problem (5), then $\hat{x} = Z\hat{z}$ is the ILS solution of the original problem (2).

The LLL algorithm first applies the Gram-Schmidt orthogonalization (GSO) to H, finding the QR factors Q and R (more precisely speaking, to avoid square root computation, the original LLL algorithm gives a column scaled Q and a row scaled R which has unit diagonal entries). Two types of basic unimodular matrices are then implicitly used to update R so that it satisfies (4): integer Gauss transformations (IGT) matrices and permutation matrices, see below.

To meet the first condition in (4), we can apply an IGT:

$$Z_{ij} = I - \zeta e_i e_j^T.$$

where e_i is the i-th column of I_n. It is easy to verify that Z_{ij} is unimodular. Applying Z_{ij} $(i < j)$ to R from the right gives

$$\bar{R} = RZ_{ij} = R - \zeta Re_i e_j^T,$$

Thus $\bar{R}$ is the same as R, except that $\bar{r}_{kj} = r_{kj} - \zeta r_{ki}$, $k = 1, \ldots, i$. By setting $\zeta = \lfloor r_{ij}/r_{ii} \rceil$, the nearest integer to r_{ij}/r_{ii}, we ensure $|\bar{r}_{ij}| \leq |\bar{r}_{ii}|/2$.

To meet the second condition in (4), we permutations columns. Suppose that we interchange columns $i - 1$ and i of R. Then the upper triangular structure of R is no longer maintained. But we can bring R back to an upper triangular matrix by using the GSO technique (see [8]):

$$\bar{R} = G_{i-1,i} R P_{i-1,i},$$

where $G_{i-1,i}$ is an orthogonal matrix and $P_{i-1,i}$ is a permutation matrix. Thus,

$$\bar{r}_{i-1,i-1}^2 = r_{i-1,i}^2 + r_{i,i}^2, \tag{6}$$
$$\bar{r}_{i-1,i}^2 + \bar{r}_{i,i}^2 = r_{i-1,i-1}^2.$$

If $\delta r_{i-1,i-1}^2 > r_{i-1,i}^2 + r_{i,i}^2$, then the above operation guarantees $\delta \bar{r}_{i-1,i-1}^2 < \bar{r}_{i-1,i}^2 + \bar{r}_{i,i}^2$.

The LLL reduction process is described in Algorithm 1.

III. SD Search Process and Babai Integer Point

For later use we briefly introduce the often used SD search process (see, e.g., [7, Section II.B.]), which is a depth-first search (DFS) through a tree. The idea of SD is to search for the optimal solution of (5) in a hyper-ellipsoid defined as follow:

$$\|\bar{y} - Rz\|_2^2 < \beta. \tag{7}$$

Define

$$c_n = \bar{y}_n/r_{nn},$$
$$c_k = \left(\bar{y}_k - \sum_{j=k+1}^{n} r_{kj}z_j\right)/r_{kk}, \quad k = n - 1, \ldots, 1. \tag{8}$$

Then it is easy to show that (7) is equivalent to

$$\text{level } k: \quad r_{kk}^2(z_k - c_k)^2 < \beta - \sum_{j=k+1}^{n} r_{jj}^2(z_j - c_j)^2, \tag{9}$$

where $k = n, n - 1, \ldots, 1$.

Algorithm 1 LLL reduction

1: apply GSO to obtain $H = QR$;
2: set $Z = I_n$, $k = 2$;
3: **while** $k \leq n$ **do**
4:    apply IGT $Z_{k-1,k}$ to reduce $r_{k-1,k}$: $R = RZ_{k-1,k}$;
5:    update Z: $Z = ZZ_{k-1,k}$;
6:    **if** $\delta r_{k-1,k-1}^2 > \left(r_{k-1,k}^2 + r_{k,k}^2\right)$ **then**
7:       permute and triangularize R: $R = G_{k-1,k}RP_{k-1,k}$;
8:       update Z: $Z = ZP_{k-1,k}$;
9:       $k = k - 1$, when $k > 2$;
10:    **else**
11:       **for** $i = k - 2, \ldots, 1$ **do**
12:          apply IGT Z_{ik} to reduce r_{ik}: $R = RZ_{ik}$;
13:          update Z: $Z = ZZ_{i,k}$;
14:       **end for**
15:       $k = k + 1$;
16:    **end if**
17: **end while**

Suppose $z_n, z_{n-1}, \ldots, z_{k+1}$ have been fixed, we try to determine z_k at level k by using (9). We first compute c_k and then take $z_k = \lfloor c_k \rceil$. If (9) holds, we move to level $k - 1$ to try to fix z_{k-1}. If at level $k - 1$, we cannot find any integer for z_{k-1} such that (9) (with k replaced by $k - 1$) holds, we move back to level k and take z_k to be the next nearest integer to c_k. If (9) holds for the chosen value of z_k, we again move to level $k - 1$; otherwise we move back to level $k + 1$, and so on. Thus after $z_n, \ldots, z_{k+1}$ are fixed, we try all possible values of z_k in the following order until (9) dose not hold anymore and we move back to level $k + 1$:

$$\begin{aligned} &\lfloor c_k \rceil, \lfloor c_k \rceil - 1, \lfloor c_k \rceil + 1, \lfloor c_k \rceil - 2, \ldots, \text{ if } c_k \leq \lfloor c_k \rceil, \\ &\lfloor c_k \rceil, \lfloor c_k \rceil + 1, \lfloor c_k \rceil - 1, \lfloor c_k \rceil + 2, \ldots, \text{ if } c_k > \lfloor c_k \rceil. \end{aligned} \tag{10}$$

When we reach level 1, we compute c_1 and take $z_1 = \lfloor c_1 \rceil$. If (9) (with $k = 1$) holds, an integer point, say $\hat{z}$, is found. We update β by setting $\beta = \|y - R\hat{z}\|_2^2$ and try to update $\hat{z}$ to find a better integer point in the new hyper-ellipsoid. Finally when we cannot find any new value for z_n at level n such that the corresponding inequality holds, the search process stops and the latest found integer point is the optimal solution we seek.

At the beginning of the search process, we set $\beta = \infty$. The first integer point z found in the search process is referred to as the Babai integer point.

IV. Partial LLL Reduction

A. Effects of size reduction on search

Ling and Howgrave-Graham [1] has argued geometrically that the performance of the Babai integer point is not affected by size reduction (see the first condition in (4)). This result can be extended. In fact we will prove algebraically that the search process is not affected by size reduction.

We stated in Section II that the size-reduction condition in (4) is met by using IGTs. It will be sufficient if we can show that one IGT will not affect the search process. Suppose that

two upper triangular matrices $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ and $\bar{\boldsymbol{R}} \in \mathbb{R}^{n \times n}$ have the relation:

$$\bar{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{Z}_{st}, \quad \boldsymbol{Z}_{st} = \boldsymbol{I} - \zeta \boldsymbol{e}_s \boldsymbol{e}_t^T, \quad s < t.$$

Thus,

$$\bar{r}_{kt} = r_{kt} - \zeta r_{ks}, \qquad \text{if } k \leq s, \tag{11}$$
$$\bar{r}_{kj} = r_{kj}, \qquad \text{if } k > s \text{ or } j \neq t. \tag{12}$$

Let $\bar{\boldsymbol{z}} = \boldsymbol{Z}_{st}^{-1}\boldsymbol{z}$. Then the ILS problem (5) is equivalent to

$$\min_{\bar{\boldsymbol{z}} \in \mathbb{Z}^n} \|\bar{\boldsymbol{y}} - \bar{\boldsymbol{R}}\bar{\boldsymbol{z}}\|_2^2. \tag{13}$$

For this ILS problem, the inequality the search process needs to check at level k is

$$\text{level } k: \quad \bar{r}_{kk}^2(\bar{z}_k - \bar{c}_k)^2 < \beta - \sum_{j=k+1}^n \bar{r}_{jj}^2(\bar{z}_j - \bar{c}_j)^2, \tag{14}$$

Now we look at the search process for the two equivalent ILS problems.

Suppose $\bar{z}_n, \bar{z}_{n-1}, \ldots, \bar{z}_{k+1}$ and $z_n, z_{n-1}, \ldots, z_{k+1}$ have been fixed. We consider the search process at level k under three different cases.

- Case 1: $k > s$. Note that $\bar{\boldsymbol{R}}_{k:n,k:n} = \boldsymbol{R}_{k:n,k:n}$. It is easy to see that we must have $\bar{c}_i = c_i$ and $\bar{z}_i = z_i$ for $i = n, n-1, \ldots, k+1$. Thus at level k, $\bar{c}_k = c_k$ and the search process takes an identical value for $\bar{z}_k$ and z_k. For the chosen value, the two inequalities (9) and (14) are identical. So both hold or fail at the same time.

- Case 2: $k = s$. According to Case 1, we have $\bar{z}_i = z_i$ for $i = n, n-1, \ldots, s+1$. Thus

$$
\begin{aligned}
\bar{c}_k &= \frac{\bar{y}_k - \sum_{j=k+1}^n \bar{r}_{kj}\bar{z}_j}{\bar{r}_{kk}} \\
&= \frac{\bar{y}_k - \sum_{j=k+1,j\neq t}^n r_{kj}z_j - (r_{kt} - \zeta r_{kk})z_t}{r_{kk}} \\
&= c_k + \zeta z_t,
\end{aligned}
$$

where ζ and z_t are integers. Note that z_k and $\bar{z}_k$ take on values according to (10). Thus values of z_k and $\bar{z}_k$ taken by the search process at level k must satisfy $\bar{z}_k = z_k + \zeta z_t$. In other words, there exists one-to-one mapping between the values of z_k and $\bar{z}_k$. For the chosen values of $\bar{z}_k$ and z_k, $\bar{z}_k - \bar{c}_k = z_k - c_k$. Thus, again the two inequalities (9) and (14) are identical. Therefore both inequalities hold or fail at the same time.

- Case 3: $k < s$. According to Case 1 and Case 2, $\bar{z}_i = z_i$ for $i = n, n-1, \ldots, s+1$ and $\bar{z}_s = z_s + \zeta z_t$. Then for $k = s - 1$,

$$
\begin{aligned}
\bar{c}_k &= \frac{\bar{y}_k - \sum_{j=k+1}^n \bar{r}_{kj}\bar{z}_j}{\bar{r}_{kk}} \\
&= \frac{\bar{y}_k - \sum_{j=k+2,j\neq t}^n r_{kj}z_j - r_{ks}\bar{z}_s - \bar{r}_{kt}z_t}{r_{kk}} \\
&= \frac{\bar{y}_k - \sum_{j=k+1}^n r_{kj}z_j - \zeta r_{ks}z_t + \zeta r_{ks}z_t}{r_{kk}} \\
&= c_k.
\end{aligned}
$$

Thus the search process takes an identical value for $\bar{z}_k$ and z_k when $k = s - 1$. By induction we can similarly show this is true for a general $k < s$. Thus, again the two inequalities (9) and (14) are identical. Therefore they hold or fail at the same time.

In the above we have proved that the search process is identical for both ILS problems (5) and (13) (actually the two search trees have an identical structure). Thus the speed of the search process is not affected by the size-reduction condition in (4). For any two integer points $\bar{\boldsymbol{z}}^*$ and $\boldsymbol{z}^*$ found in the search process at the same time for the two ILS problems, we have seen that $\bar{z}_i^* = z_i^*$ for $i = n, \ldots, s+1, s-1, \ldots, 1$ and $\bar{z}_s^* = z_s^* + \zeta z_t^*$, i.e., $\bar{\boldsymbol{z}}^* = \boldsymbol{Z}_{st}^{-1}\boldsymbol{z}^*$. Then

$$\|\bar{\boldsymbol{y}} - \bar{\boldsymbol{R}}\bar{\boldsymbol{z}}^*\|_2^2 = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}^*\|_2^2.$$

Thus, the performance of the Babai point is not affected by the size-reduction condition in (4) either, as what [1] has proved from a geometric perspective.

However, the IGTs which reduce the super-diagonal entries of $\boldsymbol{R}$ are not useless when they are followed by permutations. Suppose $|r_{i-1,i}| > \frac{|r_{i-1,i-1}|}{2}$. If we apply $\boldsymbol{Z}_{i-1,i}$ to reduce $r_{i-1,i}$, permute columns $i-1$ and i of $\boldsymbol{R}$ and triangularize it, we have from (6) and (11) that

$$
\begin{aligned}
\bar{r}_{i-1,i-1}^2 &= \left(r_{i-1,i} - \left\lfloor \frac{r_{i-1,i}}{r_{i-1,i-1}} \right\rceil r_{i-1,i-1} \right)^2 + r_{ii}^2 \\
&< r_{i-1,i}^2 + r_{ii}^2.
\end{aligned}
$$

From (6) we observe that the IGT can make $|r_{i-1,i-1}|$ smaller after permutation and triangularization. Correspondingly $|r_{i,i}|$ becomes larger, as it is easy to prove that $|r_{i-1,i-1}r_{i,i}|$ remains unchanged after the above operations.

The ELLL algorithm given in [1] is essentially identical to Algorithm 1 after lines 11–14, which reduce other off-diagonal entries of $\boldsymbol{R}$, are removed.

B. Numerical stability issue

We have shown that in the LLL reduction, an IGT is useful only if it reduces a super-diagonal entry. Theoretically, all other IGTs will have no effect on the search process. But simply removing those IGTs can causes serious numerical stability problem even $\boldsymbol{H}$ is not ill conditioned. The main cause of the stability problem is that during the reduction process, some entries of $\boldsymbol{R}$ may grow significantly. For the following $n \times n$ upper triangular matrix

$$
\boldsymbol{H} = \begin{bmatrix}
1 & 2 & 4 & & & & \\
& 1 & 2 & 0 & & & \\
& & 1 & 2 & 4 & & \\
& & & 1 & 2 & \ddots & \\
& & & & 1 & \ddots & \\
& & & & & \ddots &
\end{bmatrix}, \tag{15}
$$

when $n = 100$, the condition number $\kappa_2(\boldsymbol{H}) \approx 34$. The LLL reduction will reduce $\boldsymbol{H}$ to an identity matrix $\boldsymbol{I}$. However, if we apply the ELLL reduction, the maximum absolute value in

$\boldsymbol{R}$ will be 2^{n-1}. When n is big enough, an integer overflow will occur.

In the ELLL algorithm, the super-diagonal entries are always reduced. But if a permutation does not occur immediately after the size reduction, then this size reduction is useless in theory and furthermore it may help the growth of the other off-diagonal entries in the same column. Therefore, for efficiency and numerical stability, we propose a new strategy of applying IGTs in Algorithm 1. First we compute $\zeta = \lfloor r_{k-1,k}/r_{k-1,k-1} \rceil$. Then we test if the following inequality

$$\delta\, r_{k-1,k-1}^2 > (r_{k-1,k} - \zeta r_{k-1,k-1})^2 + r_{kk}^2$$

holds. If it does not, then the permutation of columns $k-1$ and k will not occur, no IGT will be applied, and the algorithm moves to column $k+1$. Otherwise, if $\zeta \neq 0$, the algorithm reduces $r_{k-1,k}$ and if $|\zeta| \geq 2$, the algorithm also reduces all $r_{i,k}$ for $i = k-2, k-3, \ldots, 1$ for stability consideration. When $|\zeta| = 1$, we did not notice any stability problem if we do not reduce the above size of $r_{i,k}$ for $i = k-2, k-3, \ldots, 1$.

C. Householder QR with minimum column pivoting

In the original LLL reduction and the ELLL reduction, GSO is used to compute the QR factorization of $\boldsymbol{H}$ and to update $\boldsymbol{R}$ in the later steps. The cost of computing the QR factorization by GSO is $2n^3$ flops, larger than $4n^3/3$ flops required by the QR factorization by Householder reflections (note that we do not need to form the Q factor explicitly in the reduction process); see, e.g., [10, Chap 5]. Thus we propose to compute the QR factorization by Householder reflections instead of GSO.

Roughly speaking, the reduction would like to have small diagonal entries at the beginning and large diagonal entries at the end. In our new reduction algorithm, the IGTs are applied only when a permutation will occur. The less occurrences of permutations, the faster the new reduction algorithm runs. To reduce the occurrences of permutations in the reduction process, we propose to compute the QR factorization with minimum-column-pivoting:

$$\boldsymbol{Q}^T \boldsymbol{H} \boldsymbol{P} = \boldsymbol{R} \tag{16}$$

where $\boldsymbol{P} \in \mathbb{Z}^{n \times n}$ is a permutation matrix. In the k-th step of the QR factorization, we find the column in $\boldsymbol{H}_{k:n,k:n}$, say column j, which has the minimum 2-norm. Then we interchange columns k and j of $\boldsymbol{H}$. After this we do what the k-th step of a regular Householder QR factorization does. Algorithm 2 describes the process of the factorization.

Note that the cost of computation of l_j in the algorithm is negligible compared with the other cost.

As Givens rotations have better numerical stability than GSO, in line 7 of Algorithm 1, we propose to use a Givens rotation to do triangularization.

D. PLLL reduction algorithm

Now we combine the strategies we proposed in the previous subsections and give a description of the reduction process in Algorithm 3, to be referred to as a partial LLL (PLLL) reduction algorithm.

Algorithm 2 QR with minimum-column-pivoting
1: set $\boldsymbol{R} = \boldsymbol{H}, \boldsymbol{P} = \boldsymbol{I}_n$;
2: compute $l_k = \|\boldsymbol{r}_k\|_2^2$, $k = 1 \ldots, n$;
3: **for** $k = 1, 2, \ldots, n$ **do**
4:     find j such that l_j is the minimum among $l_k, \ldots, l_n$;
5:     exchange columns k and j of $\boldsymbol{R}$, $\boldsymbol{l}$ and $\boldsymbol{P}$;
6:     apply a Householder reflection $\boldsymbol{Q}_k$ to eliminate $r_{k+1,k}, r_{k+2,k}, \ldots, r_{n,k}$;
7:     update l_j by setting $l_j = l_j - r_{k,j}^2$, $j = k+1, \ldots, m$;
8: **end for**

Algorithm 3 PLLL reduction
1: compute the Householder QR factorization with minimum pivoting: $\boldsymbol{Q}^T \boldsymbol{H} \boldsymbol{P} = \boldsymbol{R}$;
2: set $\boldsymbol{Z} = \boldsymbol{P}$, $k = 2$;
3: **while** $k \leq n$ **do**
4:     $\zeta = \lfloor r_{k-1,k}/r_{k-1,k-1} \rceil$, $\alpha = (r_{k-1,k} - \zeta r_{k-1,k-1})^2$;
5:     **if** $\delta\, r_{k-1,k-1}^2 > (\alpha + r_{k,k}^2)$ **then**
6:         **if** $\zeta \neq 0$ **then**
7:             apply the IGT $\boldsymbol{Z}_{k-1,k}$ to reduce $r_{k-1,k}$;
8:             **if** $|\zeta| \geq 2$ **then**
9:                 **for** $i = k-1, \ldots, 1$ **do**
10:                     apply the IGT $\boldsymbol{Z}_{i,k}$ to reduce $r_{i,k}$;
11:                     update $\boldsymbol{Z}$: $\boldsymbol{Z} = \boldsymbol{Z}\boldsymbol{Z}_{i,k}$;
12:                 **end for**
13:             **end if**
14:         **end if**
15:         permute and triangularize: $\boldsymbol{R} = \boldsymbol{G}_{k-1,k}\boldsymbol{R}\boldsymbol{P}_{k-1,k}$;
16:         update $\boldsymbol{Z}$: $\boldsymbol{Z} = \boldsymbol{Z}\boldsymbol{P}_{k-1,k}$;
17:         $k = k - 1$, when $k > 2$;
18:     **else**
19:         $k = k + 1$;
20:     **end if**
21: **end while**

V. NUMERICAL EXPERIMENTS

In this section we give numerical test results to compare efficiency and stability of LLL, ELLL and PLLL. Our simulations were performed in MATLAB 7.8 on a PC running Linux. The parameter δ in the reduction was set to be $3/4$ in the experiments. Two types of matrices were tested.

1) Type 1. The elements of $\boldsymbol{H}$ were drawn from an i.i.d. zero-mean, unit variance Gaussian distribution.
2) Type 2. $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the Q-factors of the QR factorizations of random matrices and $\boldsymbol{D}$ is a diagonal matrix, whose first half diagonal entries follow an i.i.d. uniform distribution over 10 to 100, and whose second half diagonal entries follow an i.i.d. uniform distribution over 0.1 to 1. So the condition number of $\boldsymbol{H}$ is bounded up by 1000.

For matrices of Type 1, we gave 200 runs for each dimension n. Figure 1 gives the average flops of the three reduction algorithms, and Figure 2 gives the average relative backward error $\|\boldsymbol{H} - \boldsymbol{Q}_c\boldsymbol{R}_c\boldsymbol{Z}_c^{-1}\|_2/\|\boldsymbol{H}\|_2$, where $\boldsymbol{Q}_c$, $\boldsymbol{R}_c$ and $\boldsymbol{Z}_c^{-1}$

Fig. 1.   Matrices of Type 1 - Flops
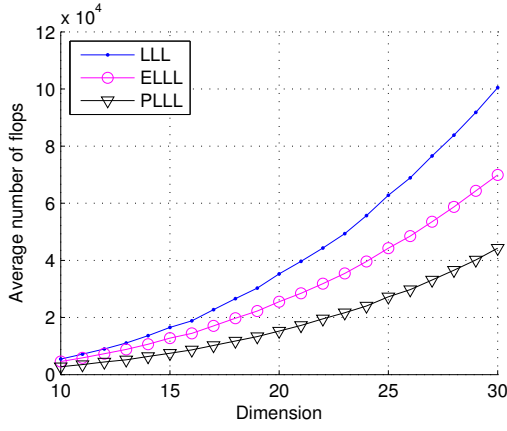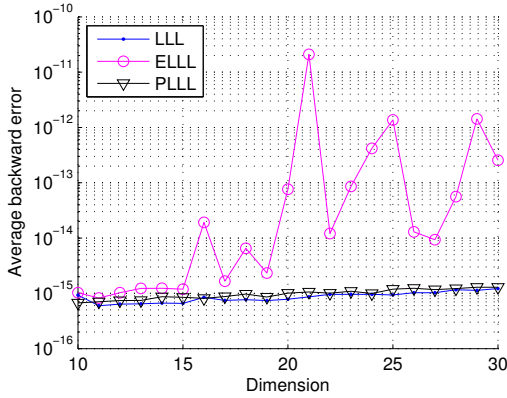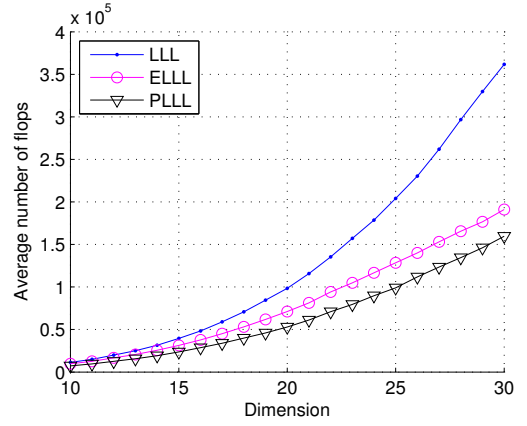

Fig. 3.   Matrices of Type 2 - Flops

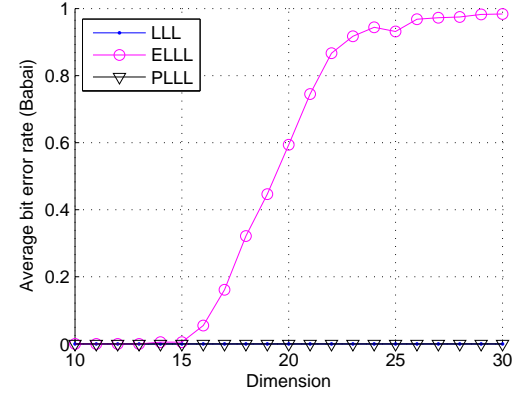Fig. 2.   Matrices of Type 1 - Backward Errors

Fig. 4.   Matrices of Type 2 - BER

are the computed factors of the QRZ factorization produced by the reduction. From Figure 1 we see that PLLL is faster than both LLL and ELLL. From Figure 2 we observe that the relative backward error for both LLL and PLLL behaves like $O(nu)$, where $u \approx 10^{-16}$ is the unit round off. Thus the two algorithms are numerically stable for these matrices. But ELLL is not numerically stable sometimes.

For matrices of Type 2, Figure 3 displays the average flops of the three reduction algorithms over 200 runs for each dimension n. Again we see that PLLL is faster than both LLL and ELLL.

To see how the reduction affects the performance of the Babai integer point, for Type 2 of matrices, we constructed the linear model $y = Hx + v$, where x is an integer vector randomly generated and $v \sim \mathcal{N}(0, 0.2^2 I)$. Figure 4 shows the average bit error rate (BER) over 200 runs for each dimension n. Form the results we observe that the computed Babai points obtained by using LLL and PLLL performed perfectly, but the computed Babai points obtained by using ELLL performed badly when the dimension n is larger than 15. Our simulations showed that the computed ILS solutions obtained by using the three reduction algorithms behaved similarly. All these indicate that ELLL can give a very poor estimate of x due to its numerical stability problem.

REFERENCES

[1] C. Ling and N. Howgrave-Graham, "Effective lll reduction for lattice decoding," in *Proc. IEEE International Symposium on Information Theory*, 2007, pp. 196–200.

[2] P. v. Boas, "Another NP-complete partition problem and the complexity of computing short vectors in a lattice," Mathematisch Institute, Amsterdam, The Netherlands, Tech. Rep. 81-04, 1981.

[3] L. Babai, "On lovasz's lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.

[4] C. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[5] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201–2214, 2002.

[6] M. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[7] X.-W. Chang and Q. Han, "Solving box-constrained integer least-squares problems," *IEEE Transactions on Wireless Communications*, vol. 7, no. 1, pp. 277–287, 2008.

[8] A. Lenstra, J. Lenstra, and L. Lovasz, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, pp. 515–534, 1982.

[9] X.-W. Chang and G. Golub, "Solving ellipsoid-constrained integer least-squares problems," *SIAM J. Matrix Anal. Appl.*, vol. 31, pp. 1071–1089, 2009.

[10] G. Golub and C. Van Loan, *Matrix Computations*.   The Johns Hopkins University Press, Baltimore, Maryland, 3rd edition, 1996.